OPTIMIZATION OF MODEL-FREE ADAPTIVE CONTROLLER USING DIFFERENTIAL EVOLUTION METHOD

Leandro dos Santos Coelho
Industrial and Systems Engineering Graduate Program, PPGEPS
Pontifical Catholic University of Paraná
Rua Imaculada Conceição, 1155
80215-901, Curitiba, Paraná, Brazil
leandro.coelho@pucpr.br

Abstract*. It is well-known that conventional control theories are widely suited for applications where the processes can be reasonably described in advance. However, when the plant's dynamics are hard to characterize precisely or are subject to environmental uncertainties, one may encounter difficulties in applying the conventional controller design methodologies. Despite the difficulty in achieving high control performance, the fine tuning of controller parameters is a tedious task that always requires experts with knowledge in both control theory and process information. Nowadays, more and more studies have focused on the development of adaptive control algorithms that can be directly applied to complex processes whose dynamics are poorly modeled and/or have severe nonlinearities. In this context, the design of a Model-Free Learning Adaptive Control (MFLAC) based on pseudo-gradient concepts and optimization procedure by Differential Evolution (DE) is presented in this paper. DE algorithms are evolutionary algorithms that have already shown appealing features as efficient methods for the optimization of continuous space functions. Motivation for application of DE approach is to overcome the limitation of the conventional MFLAC design, which cannot guarantee satisfactory control performance when the plant has different gains for the operational range when designed by trial-and-error by user. Numerical results of the MFLAC with particle swarm optimization for a nonlinear control valve are showed.*

Keywords*: adaptive control, model-free adaptive control, differential evolution.*

1. INTRODUCTION

Model-based control techniques are usually implemented under the assumption of good understanding of process dynamics and their operational environment. These techniques, however, cannot provide satisfactory results when applied to poorly modeled processes, which can operate in ill-defined environments. This is often the case when dealing with complex dynamic systems for which the physical processes are either highly nonlinear or are not fully understood (Karray *et al*., 2002).

The conventional Proportional-Integral-Derivative (PID) algorithm is still widely used in process industries because its simplicity and robustness. PID controllers are the most common controllers in industry. In fact, 95% of control loops use PID and the majority is PI control (Åström and Hägglund, 2005). Consequently, many different methods have been proposed for determining the three controller parameters, i.e., proportional, integral, and derivative gains, to meet different requirements of various control applications. However, its performance is not adequate in many chemical processes. A change in the signal and the directionality of the process gain is a complex practical situation and, so, still becoming complex the design of a control system (Bisowarno *et al*., 2003).

It is believed that a fixed-parameter PID may not do well for nonlinear, time-variant, or coupled processes. It needs to be re-tuned adequately to retain robust control performance over a wide range of operating conditions (Ali, 2000). Alternatively, several approaches have been proposed in the literature for controlling nonlinear processes, such as model predictive control, neural control, fuzzy control, robust control, sliding mode control, and adaptive control.

Adaptive control methods are able to cope with control problems involving internal process uncertainties as well as external environmental uncertainties. The aim of this paper is to merge for nonlinear systems, the model-free learning adaptive control structure (Hou and Huang, 1997; Hou *et al*., 1998) with the controller design optimization based on Differential Evolution (DE). DE is an evolutionary algorithm originally proposed by Storn and Price (1995) and Storn (1997) whose main design emphasis is real parameter optimization. DE is based on a mutation operator, which adds an amount obtained by the difference of two randomly chosen individuals of the current population, in contrast to most evolutionary algorithms, in which the mutation operator is defined by a probability function. Despite DE's apparent simplicity, the interacting key evolutionary operators of mutation and recombination are present and effective. In particular, DE has the advantage of incorporating a relatively simple and efficient form of self-adapting mutation.

The remainder of this paper is organized as follows. In section 2, a model-free learning adaptive control structure is described, while section 3 explains the concepts of DE optimization method. Section 4 presents the simulation results for the control of a valve with nonlinear behaviour. Lastly, section 5 outlines our conclusion and future research.

2. MODEL-FREE LEARNING ADAPTIVE CONTROL

It is well-known that conventional control theories are widely suited for applications where the processes can be reasonably described in advance. However, when the plant's dynamics are hard to characterize precisely or are subject to environmental uncertainties, one may encounter difficulties in applying the conventional controller design methodologies.

Adaptive control schemes are alternatives for handling processes with unknown nonlinearities. In the past few decades, there has been considerable interest in the development of adaptive control systems that automatically adjust controller parameters to compensate for unanticipated changes in the process and/or the environment. The ability of dealing with time-varying characteristics, nonlinearities, and uncertainties enables adaptive control algorithms to have significant potential for the operation of complex processes whose dynamics are imprecisely known and/or are subject to changes in unpredictable ways.

Closed-loop adaptive control methods may be divided into two broad categories: (i) indirect or explicit control, (ii) direct or implicit control. Indirect control methods utilize separate parameter identification and control schemes. The plant parameters are estimated explicitly on-line and the control parameters are then adjusted based on these estimations. In contrast to this, direct control methods do not utilize an explicit identification of the process parameters. The identification and control functions are merged into one scheme. The controller parameters are adjusted directly, only using plant input and output signals (Ozcelik and Kaufman, 1999).

In this paper, a direct adaptive control of the following general discrete SISO (Single-Input and Single-Output) nonlinear system is considered

$$y(k+1) = f\left(y(k),\cdots,y(k-n_a),u(k),\cdots,u(k-n_b)\right) \tag{1}$$

where n_a and n_b are the orders of system output, $y(k)$, and input, $u(k)$, respectively, and $f(\cdot)$ is a general nonlinear function. The plant (equation 1) can be rewritten as follows:

$$y(k+1) = f\left(Y(k),u(k),U(k-1)\right) \tag{2}$$

where $Y(k)$ and $U(k\text{-}1)$ are the sets of system outputs and inputs up to sampling instant k and k-1.

The following assumptions are considered about the controlled plant: (A1) the system (1) and (2) is observable and controllable; (A2) the partial derivative of $f(\cdot)$ with respect to control input $u(k)$ is continuous; and (A3) the system (1) is generalized Lipschitz.

For a nonlinear system (2), satisfying assumptions (A1-A3), then there must exist $\phi(k)$, called pseudo-gradient vector, when control change $\Delta u(k) \neq 0$, and

$$\Delta y(k+1) = f^T(k)\Delta u(k) \tag{3}$$

where the control change $\Delta u(k) = u(k) - u(k\text{-}1)$; $\left\|\phi(k)\right\| \leq L,$ and L is a constant.

Details of the theoretical basis and the mathematical proof of the MFLAC are given in Hou and Huang (1997) and Hou *et al.* (1998). In this proof, the equation $y(k+1) = f\left(Y(k),u(k),U(k-1)\right)$ gives

$$\Delta y(k+1) = f\left(Y(k),u(k),U(k-1)\right) - f\left(Y(k-1),u(k-1),U(k-2)\right) \tag{4}$$

or

$$\begin{aligned}\Delta y(k+1) = {} & f\left(Y(k),u(k),U(k-1)\right) - f\left(Y(k),u(k-1),U(k-1)\right) + \\ & f\left(Y(k),u(k),U(k-1)\right) - f\left(Y(k-1),u(k-1),U(k-2)\right)\end{aligned} \tag{5}$$

Using assumption (A2) and the mean value theorem, equation (5) gives

$$\Delta y(k+1) = \frac{\partial f^-}{\partial u(k)}\Delta u(k) + x(k) \tag{6}$$

where $\frac{\partial f^-}{\partial u(k)}$ denotes the value of gradient vector of $f(Y(k),u(k),U(k-1))$ with respect to u at some point between $u(k-1)$ and $u(k)$, and $x(k)$ given by

$$\xi(k) = f(Y(k),u(k-1),U(k-1)) - f(Y(k-1),u(k-1),U(k-2)) \tag{7}$$

Considering the following equation

$$x(k) = h^T(k)\Delta u(k) \tag{8}$$

where $h(k)$ is a variable. Since condition $\Delta u(k) \neq 0,$ equation (8) must have solution $h(k)$. Let

$$f(k) = \frac{\partial f^-}{\partial u(k)} + h(k) \tag{9}$$

From (8) and (9), then (7) can be rewritten as $\Delta y(k+1) = f^T(k)\Delta u(k)$. This is the same as (3). In this case, by using (3) and assumption (A3), and $\Delta u(k) \neq 0,$ we have

$$\left| f^T(k)\Delta u(k) \right| \leq L \|\Delta u(k)\| \tag{10}$$

Hence $\|f(k)\| \leq L$. For the learning control law algorithm, a weighted one-step-ahead control input cost function is adopted, and given by

$$J(u(k)) = [y(k+1) - y_r(k+1)]^2 + \lambda \|\Delta u(k)\|^2 \tag{11}$$

For the control design, where $y_r(k+1)$ is the expected system output signal (true output of the controlled plant), and λ is a positive weighted constant. The equation (3) can be rewrite as follows

$$y(k+1) = y(k) + \phi^T(k)\Delta u(k) \tag{12}$$

Substituting (12) into (11), differentiating (11) with respect to $u(k)$, solving the equation $\partial J(u(k))/\partial u(k) = 0$, and using the matrix-inversion-lemma gives the control law as follows:

$$u(k) = u(k-1) + \frac{\rho_k \phi(k)}{\lambda + \|\phi(k)\|^2}[y_r(k+1) - y(k)] \tag{13}$$

The control law (13) is a kind of control that has no relationship with any structural information (mathematical model, order, structure, etc.) of the controlled plant. It is designed only using I/O data of the plant.

The cost function proposed by Hou *et al.* [5] for parameter estimation is used in this paper as

$$J(\phi(k)) = \left[y(k) - y(k-1) - \phi^T \Delta u(k-1)\right]^2 + \mu \left\|\phi(k) - \hat{\phi}(k-1)\right\|^2 \tag{14}$$

Using the similar procedure of control law equations, we can obtain the parameter estimation algorithm as follows:

$$\hat{\phi}(k) = \hat{\phi}(k-1) + \frac{\eta \Delta u(k-1)}{\mu + \|\Delta u(k)\|^2}\left[\Delta y(k) - \hat{\phi}^T(k-1)\Delta u(k-1)\right] \tag{15}$$

Summarizing, the MFLAC scheme is

$$\hat{\phi}(k) = \hat{\phi}(k-1) + \frac{\eta \Delta u(k-1)}{\mu + \|\Delta u(k)\|^2}\left[\Delta y(k) - \hat{\phi}^T(k-1)\Delta u(k-1)\right] \tag{16}$$

$\hat{\phi}(k) = \hat{\phi}(1)$ if

$$sign(\phi(1)) \neq sign(\hat{\phi}(k)) \tag{17}$$

$\hat{\phi}(k) = \hat{\phi}(1)$ if

$$\left\|\hat{f}(k)\right\| \geq M, \text{ or } \left\|\hat{f}(k)\right\| \leq e \tag{18}$$

$$u(k) = u(k-1) + \frac{\rho_k \phi(k)}{\lambda + \left\|\phi(k)\right\|^2}\left[y_r(k+1) - y(k)\right] \tag{19}$$

where step-size series ρ and η, and the weighted constants λ and μ are design parameters optimized by differential evolution in this paper. The parameter ε is a small positive constant (adopted 0.00001), M is adopted with value 10, and $\hat{\phi}(k) = \hat{\phi}(1)$ is the initial estimation value of $\phi(k)$.

3. OPTIMIZATION OF MFLAC DESIGN USING DIFFERENTIAL EVOLUTION

Evolutionary algorithms are a broad class of stochastic optimization algorithms inspired by biology and, in particular, by those biological processes that allow populations of organisms to adapt to their surrounding environments: genetic inheritance and survival of the fittest. Evolutionary algorithms have a prominent advantage over other types of numerical methods. They only require information about the objective function itself, which can be either explicit or implicit (Brest *et al.*, 2007).

The DE algorithm (Storn and Price, 1995; Storn, 1997) is an evolutionary algorithm which uses a rather greedy and less stochastic approach to problem solving than do classical evolutionary algorithms such as genetic algorithms, evolutionary programming, and evolution strategies. DE also incorporates an efficient way of self-adapting mutation using small populations.

DE is a very simple but very powerful stochastic global optimizer. The crucial idea behind DE is a scheme for generating trial parameter vectors. Like all genetic algorithms, DE is population based. It evolutes generation by generation until the termination conditions have been met (Qing, 2006).

The different variants of DE are classified using the following notation: DE/*a*/*b*/*d*, where *a* indicates the method for selecting the parent chromosome that will form the base of the mutated vector, *b* indicates the number of difference vectors used to perturb the base chromosome, and *d* indicates the recombination mechanism used to create the offspring population. The *bin* acronym indicates that the recombination is controlled by a series of independent binomial experiments. The variant implemented here of DE was the DE/*rand*/1/*bin*, which involved the following steps:

Step 1: *Parameter setup*

The user chooses the parameters of population size, the boundary constraints of optimization variables, the mutation factor (f_m), the crossover rate (*CR*), and the stopping criterion of maximum number of iterations (generations), t_{max}.

Step 2: *Initialization of the population*

Set generation t=0. Initialize a population of i=1,..,M individuals (real-valued n-dimensional solution vectors) with random values generated according to a uniform probability distribution in the n dimensional problem space. These initial individual values are chosen at random from within user-defined bounds (boundary constraints).

Step 3: *Evaluation of the population*

Evaluate the fitness value of each individual.

Step 4: *Mutation operation (or differential operation)*

Mutation is an operation that adds a vector differential to a population vector of individuals according to the following equation:

$$z_i(t+1) = x_{i,r_1}(t) + f_m \cdot [x_{i,r_2}(t) - x_{i,r_3}(t)] \tag{20}$$

where i =1,2,...,M is the individual's index of population; j=1,2,..., n is the position in n dimensional individual; t is the time (generation); $x_i(t) = \left[x_{i_1}(t), x_{i_2}(t), ..., x_{i_n}(t)\right]^T$ stands for the position of the i-th individual of population of N real-valued n-dimensional vectors; $z_i(t) = \left[z_{i_1}(t), z_{i_2}(t), ..., z_{i_n}(t)\right]^T$ stands for the position of the i-th individual of a *mutant vector*; r_1, r_2 and r_3 are mutually different integers and also different from the running index, i, randomly selected with uniform distribution from the set $\{1, 2, \cdots, i-1, i+1, \cdots, N\}$; $f_m > 0$ is a real parameter called *mutation factor*, which controls the amplification of the difference between two individuals so as to avoid search stagnation and is usually taken from the range [0.1, 1].

Step 5: *Recombination (crossover) operation*

Following the mutation operation, recombination is applied to the population. Recombination is employed to generate a trial vector by replacing certain parameters of the target vector with the corresponding parameters of a randomly generated donor vector.

For each vector, $z_i(t+1)$, an index $rnbr(i) \in \{1, 2, \cdots, n\}$ is randomly chosen using uniform distribution, and a *trial vector*, $u_i(t+1) = \left[u_{i_1}(t+1), u_{i_2}(t+1), ..., u_{i_n}(t+1)\right]^T$, is generated with

$$u_{i_j}(t+1) = \begin{cases} z_{i_j}(t+1), \text{ if } randb(j) \leq CR) \text{ or } j = rnbr(i), \\ x_{i_j}(t), \text{ if } randb(j) > CR) \text{ or } j \neq rnbr(i) \end{cases} \tag{21}$$

In the above equations, *randb(j)* is the *j*-th evaluation of a uniform random number generation with [0, 1] and *CR* is a *crossover* or *recombination rate* in the range [0, 1]. The performance of a DE algorithm usually depends on three variables: the population size *N*, the mutation factor f_m, and the recombination rate *CR*.

Step 6: *Selection operation*

Selection is the procedure of producing better offspring. To decide whether or not the vector $u_i(t+1)$ should be a member of the population comprising the next generation, it is compared with the corresponding vector $x_i(t)$. Thus, if f denotes the objective function under minimization, then

$$x_i(t+1) = \begin{cases} u_i(t+1), \text{ if } \quad f(u(t+1)) < f(x_i(t)), \\ x_i(t), \text{ otherwise} \end{cases} \tag{22}$$

In this case, the cost of each trial vector $u_i(t+1)$ is compared with that of its parent target vector $x_i(t)$. If the cost, f, of the target vector $x_i(t)$ is lower than that of the trial vector, the target is allowed to advance to the next generation. Otherwise, the target vector is replaced by the trial vector in the next generation [26].

Step 7: *Verification of stopping criterion*

Set the generation number for $t = t + 1$. Proceed to Step 3 until a stopping criterion is met, usually t_{max}. The stopping criterion depends on the type of problem.

In this paper, the DE optimization technique is adopted to obtain $f(1)$, ρ, η, λ and μ for the MFLAC design. The setup of DE applied in this work was the following:

- population size: $M = 20$;
- crossover rate: $CR = 0.8$;
- mutation factor: $f_m = 0.5$;
- stopping criterion: $t_{max} = 50$ generations.

The objective of the DE in the MFLAC optimization is to maximize the fitness equation given by

$$f = \frac{x}{1 + \left\{ \sum_{i=1}^{t} \left| y(k) - y_r(k) \right| + 0.001 \left[u(k) - u(k-1) \right]^2 \right\}} \tag{23}$$

where $u(k)$ is the control signal, $y(k)$ is the process output, and $y_r(k)$ is the reference (setpoint), and x is a scale factor (adopted $x = 0.3$).

4. SIMULATION RESULTS

The control valve system is an opening with adjustable area. Normally it consists of an actuator, a valve body and a valve plug. The actuator is a device that transforms the control signal to movement of the stem and valve plug. Wigren (1993) describes the plant where the control valve dynamic is described by a Wiener model (the nonlinear element follows linear block) and it is given by

$$x(k) = 1.5714 \cdot x(k-1) + 0.6873 \cdot x(k-2) + 0.0616 \cdot u(k\text{-}1) + 0.0543 \cdot u(k\text{-}2) \tag{24}$$

$$y(k) = f_n[x(k)] = \frac{x(k)}{\sqrt{0{,}10 + 0.90[x(k)]^2}} \tag{25}$$

where $u(k)$ is the control pressure, $x(k)$ is the stem position, and $y(k)$ is the flow through the valve which is the controlled variable. The input to the process, $u(k)$, is constrained between [0; 1.2]. The nonlinear behavior of the control valve described by equation (25) is shown in Figure 1.

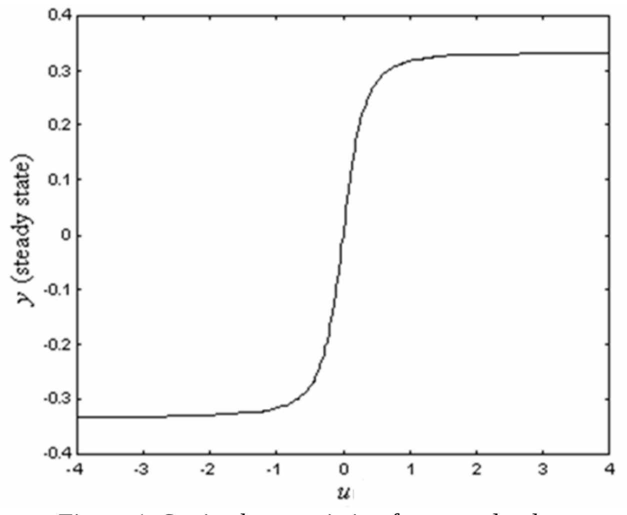

Figure 1. Static characteristic of a control valve.

The space search adopted in DE setup is: $0 \le f(1) \le 0.50$, $0 \le r \le 1$, $0 \le h \le 2$, $0 \le l \le 1.00$, and $0 \le m \le 1$. In Table 1 is presented the convergence of DE for the design of MFLAC in 30 runs. An analysis of Table 1 reveals that DE found mean fitness value closed to the best fitness value and also with small standard deviation value.

Table 1. Simulation results for MFLAC's tuning (analysis of best fitness of each run) in 30 runs.

fitness function, *f*			
best	mean	minimum	standard deviation
0.5102	0.4605	0.4079	0.0338

For the MFLAC design, the optimization procedure by DE obtains $f(1) = 0.880843$, $r = 0.237826$, $h = 1.965818$, $l = 0.119052$, and $m = 2 \cdot 10^{-16}$ with fitness $f = 0.5102$ (best result of Table 1 in 30 runs).

Results for servo and regulatory responses of MFLAC are shown in Figures 2 and 3, respectively. Regulatory behavior analysis of the MFLAC was based on parametric changes (disturbances) in the plant output when: (i) sample 10: $y(k) = y(k) + 0.05$; (ii) sample 140: $y(k) = y(k) - 0.3$; (iii) sample 260: $y(k) = y(k) – 0.2$; (iv) sample; and (iv) sample 320: $y(k) = y(k) + 0.1$.

Simulation results presented in Figures 2 and 3 show that the MFLAC using DE approach have good control performance. The nonlinear nature of control valve implies changes of the parameters due to the different operating conditions or operating point. In this case, the MFLAC was designed to follow the changes of setpoint using DE.

Performance of MFLAC design was affected by nonlinearity of control valve. Furthermore, the MFLAC design obtained fast response, reasonable control activity, and good setpoint tracking ability. The good performance indicated by the MFLAC using DE approach confirms the usefulness and robustness of the proposed method for practical applications. Although the simulation results had shown only the step responses to reference changes, good disturbance rejection properties can be obtained using MFLAC.

In Table 2, a summary of simulation results and performance of the MFLAC design based on DE is presented.

Table 2. Performance indices for the best MFLAC design using DE optimization.

MFLAC performance	servo behavior	regulatory behavior
mean of *u*	0.4630	0.4663
variance of *u*	0.0753	0.0770
mean of error	0.0024	0.0016
variance of error	0.0037	0.0045

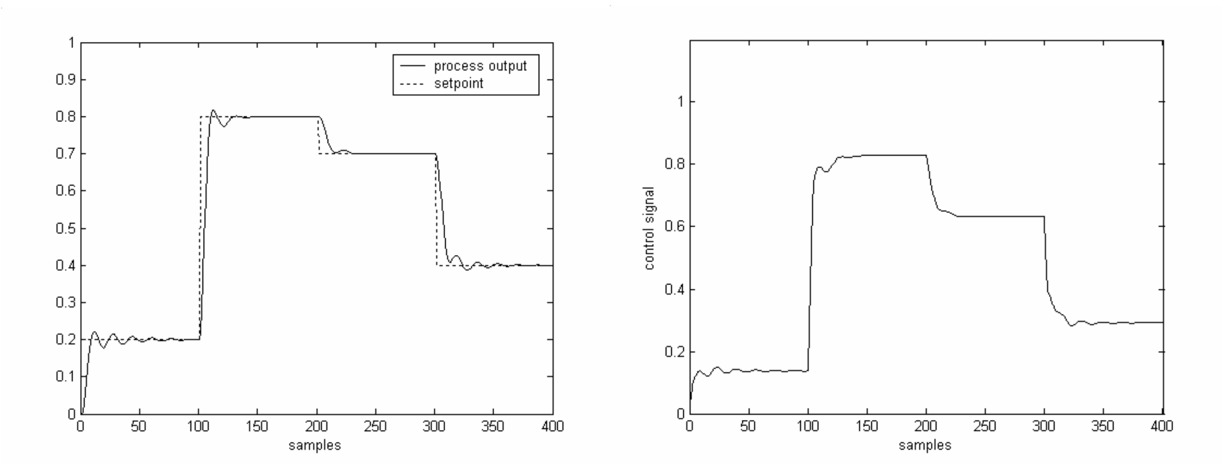

Figure 2. Input and output signals for the MFLAC (servo behavior).

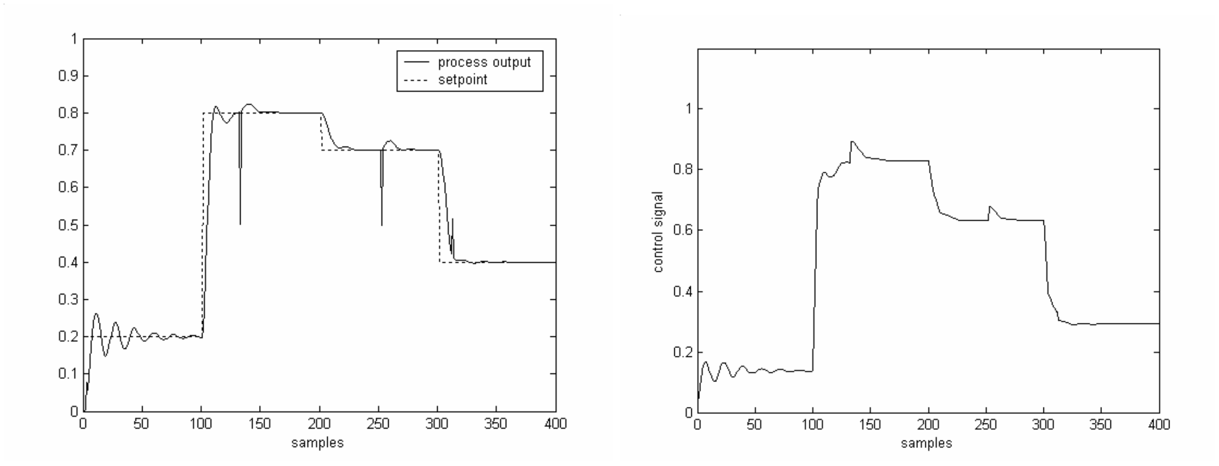

Figure 3. Input and output signals for the MFLAC (regulatory behavior).

5. CONCLUSION AND FUTURE RESEARCH

The DE algorithm is a direct method and it has a role to play where the gradient of the function is not available. Design of DE certainly brought a new dimension to the direct search techniques in the field of global optimization. In this paper, numerical results for controlling a control valve have shown the efficiency of a MFLAC design using DE optimization algorithm that guaranteed the convergence of the tracking error for servo and regulatory responses.

However, it still has a distance to industrial applications and more practical issues must be done. A further investigation can be directed to analyze the DE for model-free adaptive control methods (Spall and Cristion, 1998) in essential control issues such as control performance, robustness and stability in multivariable processes.

6. ACKNOWLEDGEMENTS

This work was supported by the National Council of Scientific and Technologic Development of Brazil — CNPq — under Grant 309646/2006-5/PQ.

7. REFERENCES

Ali, E., 2000, “Control of Nonlinear Chemical Processes Using Adaptive Proportional-Integral Algorithms”, Ind. Eng. Chem. Res., Vol. 39, pp. 1980-1992.

Åström, K.J. and Hägglund, T., 1995, “PID Controllers: Theory, Design, and Tuning”, Instrument Society of America, ISA, Triangle Park, North Carolina, USA.

Bisowarno, B.H., Tian, Y.-C. and Tade, M.O., 2003, “Model Gain Scheduling Control of an Ethyl Tert-butyl Ether Reactive Distillation Column”, Ind. Eng. Chem. Res., vol. 42, pp. 3584-3391.

Brest, J., Greiner, S., Boskovic, B., Mernik, M. and Zumer, V., 2007, “Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems”, IEEE Transactions on Evolutionary Computation (accepted for future publication).

Hou, Z. and Huang, W., 1997, “The Model-Free Learning Adaptive Control of a Class of SISO Nonlinear Systems”, Proceedings of the American Control Conference, Albuquerque, NM, USA, pp. 343-344.

Hou, Z., Han, C. and Huang, W., 1998, “The Model-Free Learning Adaptive Control of a Class of MISO Nonlinear Discrete-Time Systems”, IFAC Low Cost Automation, Shenyang, P. R. China, pp. 227-232.

Karray, F., Gueaieb, W. and Al-Sharhan, S., 2002, “The Hierarchical Expert Tuning of PID Controllers Using Tools of Soft Computing”, IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics, Vol. 32, No. 1, pp. 77-90.

Ozcelik, S. and Kaufman, H., 1999, “Design of Robust Direct Adaptive Controllers for SISO Systems: Time and Frequency Domain Design Conditions”, International Journal of Control, Vol. 72, No. 6, pp. 517-530.

Qing, A., 2006, “Dynamic Differential Evolution Strategy and Applications in Electromagnetic Inverse Scattering Problems”, IEEE Transactions on Geoscience and Remote Sensing, Vol. 44, No. 1, pp. 116-125.

Spall, J.C. and Cristion, J.A., 1998, “Model-Free Control of Nonlinear Systems with Discrete Time Measurements,” IEEE Transactions on Automatic Control, Vol. 43, pp. 1198-1210.

Storn, R. and Price, K., 1995, “Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces”, Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA, USA.

Storn, R., 1997, “Differential Evolution — A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces”, Journal of Global Optimization, Vol. 11, No. 4, pp. 341-359.

Wigren, T., 1993, “Recursive Prediction Error Identification Using The Nonlinear Wiener Model,” Automatica, Vol. 29, No. 4, pp. 1011-1025.

8. RESPONSIBILITY NOTICE